Kotlin - Data Types

Kotlin data type is a classification of data which tells the compiler how the programmer intends to use the data. For example, Kotlin data could be numeric, string, boolean etc.

Kotlin treats everything as an object which means that we can call member functions and properties on any variable.

Kotlin is a statically typed language, which means that the data type of every expression should be known at compile time.

Kotlin built in data type can be categorized as follows:

- Number
- Character
- String
- Boolean
- Array

(a) Kotlin Number Data Types

Kotlin number data types are used to define variables which hold numeric values and they are divided into two groups: (a) **Integer types** store whole numbers, positive or negative (b) **Floating point** types represent numbers with a fractional part, containing one or more decimals.

Following table list down all the Kotlin number data types, keywords to define their variable types, size of the memory taken by the variables, and a value range which can be stored in those variables.

Data Type	Size (bits)	Data Range
Byte	8 bit	-128 to 127
Short	16 bit	-32768 to 32767
Int	32 bit	-2,147,483,648 to 2,147,483,647
Long	64 bit	-9,223,372,036,854,775,808 to +9,223,372,036,854,775,807
Float	32 bit	1.40129846432481707e-45 to 3.40282346638528860e+38
Double	64 bit	4.94065645841246544e-324 to 1.79769313486231570e+308

If we will try to store a value more than permitted value in a variable of particular data type, the Kotlin compiler will complain because an overflow would occur at runtime.

Example

Following example shows how to define and access different Kotlin number data types:

```
fun main(args: Array<String>) {
   val a: Int = 10000
   val d: Double = 100.00
   val f: Float = 100.00f
   val l: Long = 1000000004
   val s: Short = 10
   val b: Byte = 1

   println("Int Value is " + a)
   println("Double  Value is " + d)
   println("Float Value is " + f)
   println("Long Value is " + l )
   println("Short Value is " + s)
   println("Byte Value is " + b)
}
```

When you run the above Kotlin program, it will generate the following output:

```
Int Value is 10000
Double Value is 100.0
Float Value is 100.0
Long Value is 1000000004
Short Value is 10
Byte Value is 1
```

(b) Kotlin Character Data Type

Kotlin character data type is used to store a single character and they are represented by the type **Char** keyword. A Char value must be surrounded by single quotes, like 'A' or '1'.

Example

Following example shows how to define and access a Kotlin Char data type:

```
fun main(args: Array<String>) {
   val letter: Char    // defining a Char variable
   letter = 'A'        // Assigning a value to it
   println("$letter")
}
```

When you run the above Kotlin program, it will generate the following output:

```
A
```

Kotlin supports a number of escape sequences of characters. When a character is preceded by a backslash (\), it is called an escape sequence and it has a special meaning to the compiler. For example, \n in the following statement is a valid character and it is called a new line character

```
println('\n') //prints a newline character

println('\$') //prints a dollar $ character

println('\\') //prints a back slash \ character
```

The following escape sequences are supported in Kotlin: \t, \b, \n, \r, \', \", \\ and \$.

(c) Kotlin String Data Type

The String data type is used to store a sequence of characters. String values must be surrounded by double quotes (" ") or triple quote (""" """).

We have two kinds of string available in Kotlin - one is called **Escaped String** and another is called **Raw String**.

- **Escaped string** is declared within double quote (" ") and may contain escape characters like '\n', '\t', '\b' etc.
- **Raw string** is declared within triple quote (""" """) and may contain multiple lines of text without any escape characters.

```
fun main(args: Array<String>) {
   val escapedString : String  = "I am escaped String!\n"
   var rawString :String  = """This is going to be a
   multi-line string and will
   not have any escape sequence""";

   print(escapedString)
   println(rawString)
}
```

When you run the above Kotlin program, it will generate the following output:

```
I am escaped String!
This is going to be a
multi-line string and will
not have any escape sequence
```

(d) Kotlin Boolean Data Type

Boolean is very simple like other programming languages. We have only two values for Boolean data type - either **true** or **false**.

```
fun main(args: Array<String>) {
   val A: Boolean = true   // defining a variable with true value
   val B: Boolean = false   // defining a variable with false value

   println("Value of variable A "+ A )
   println("Value of variable B "+ B )
}
```

When you run the above Kotlin program, it will generate the following output:

```
Value of variable A true
Value of variable B false
```

Boolean has a nullable counterpart **Boolean?** that can store a **null** value as below:

```
val boolNull: Boolean? = null
```

(e) Kotlin Array Data Type

Kotlin arrays are a collection of homogeneous data. Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.

We will study array in a separate chapter, for now let's look at one example to define an array of integers and then access its one of the elements.

```
fun main(args: Array<String>) {
   val numbers: IntArray = intArrayOf(1, 2, 3, 4, 5)
   println("Value at 3rd position : " + numbers[2])
}
```

When you run the above Kotlin program, it will generate the following output:

```
Value at 3rd position : 3
```

Kotlin Data Type Conversion

Type conversion is a process in which the value of one data type is converted into another type. Kotlin does not support direct conversion of one numeric data type to another, For example, it is not possible to convert an Int type to a Long type:

```
fun main(args: Array<String>) {
   val x: Int = 100
   val y: Long = x  // Not valid assignment

   println(y)
}
```

When you run the above Kotlin program, it will generate the following output:

```
main.kt:3:18: error: type mismatch: inferred type is Int but Long was expected
val y: Long = x // Not valid assignment
```

To convert a numeric data type to another type, Kotlin provides a set of functions:

- toByte()
- toShort()
- toInt()
- toLong()
- toFloat()
- toDouble()
- toChar()

Now let's rewrite above example once again and try to run it:

```
fun main(args: Array<String>) {
   val x: Int = 100
   val y: Long = x.toLong()

   println(y)
}
```

When you run the above Kotlin program, it will generate the following output:

```
100
```

Quiz Time (Interview & Exams Preparation)

Q 1 - Can a Kotlin Boolean variable have a value of 0:

A - True

B - False

Q 2 - Which of the data type is used to define a variable PI to store the value 3.14

A - Float

B - Double

C - Long

D - None of the above

Q 3 - What could be the minimum length of Kotlin String?

A - Zero character

B - 1 character

C - Null value

D - None of the above

Q 4 - Which of these are valid data types to hold integers in Kotlin?

A - Byte, Short, Int, Long

B - Byte, Int, Float, Long

C - Short, Int, Long, Boolean

D - None of the above

Q 5 - Which of the following statement is true about Kotlin Data Type:

A - Kotlin does not allow direct numeric data type conversion.

B - To convert a numeric data type to another type, you must use provided helper functions.

C - Int, Short, Float, Byte, Long, Double can be used to define Numeric Data types

D - All of the above